The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth

The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth

Jamie Watson[1] Oisin Mac Aodha[2] Victor Prisacariu[1,3] Gabriel Brostow[1,4] Michael Firman[1]
[1]Niantic [2]University of Edinburgh [3]University of Oxford [4]UCL
www.github.com/nianticlabs/manydepth

Abstract

Self-supervised monocular depth estimation networks are trained to predict scene depth using nearby frames as a supervision signal during training. However, for many applications, sequence information in the form of video frames is also available at test time. The vast majority of monocular networks do not make use of this extra signal, thus ignoring valuable information that could be used to improve the predicted depth. Those that do, either use computationally expensive test-time refinement techniques or off-the-shelf recurrent networks, which only indirectly make use of the geometric information that is inherently available.

We propose ManyDepth, an adaptive approach to dense depth estimation that can make use of sequence information at test time, when it is available. Taking inspiration from multi-view stereo, we propose a deep end-to-end cost volume based approach that is trained using self-supervision only. We present a novel consistency loss that encourages the network to ignore the cost volume when it is deemed unreliable, e.g. in the case of moving objects, and an augmentation scheme to cope with static cameras. Our detailed experiments on both KITTI and Cityscapes show that we outperform all published self-supervised baselines, including those that use single or multiple frames at test time.

1. Introduction

Knowing the depth to each pixel in an image has proved to be a useful and versatile tool, with applications ranging from augmented reality [59], autonomous driving [22], through to 3D reconstruction [64]. While specialist hardware can give per-pixel depth, e.g. from structured light or Lidar sensors, a more attractive approach is to only require a single RGB camera. Many recent monocular depth from RGB methods are trained using only self-supervision, which removes the need for expensive hardware to capture training depth data [21, 99, 23, 26]. While these approaches appear to be very promising, their test-time depth estimation performance is not yet on a par with specialist depth hardware or deep multi-view methods [73].

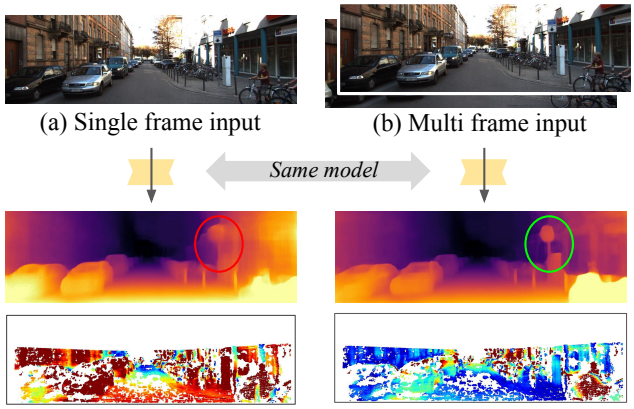

Figure 1. Trained using only self-supervision, our model not only predicts depth from single frames (a) but can also utilize multiple frames, when they are available, using the same model (b). This results in superior depth predictions at test time. Error maps on the bottom row show large depth errors as red, small as blue.

In an attempt to close this performance gap, we observe that in most practical scenarios more than one frame is available at test time e.g. from a camera on a moving vehicle or micro-baseline frames from a phone camera [32, 41]. Yet these additional frames are typically not exploited by current monocular methods. In this work, we use these additional frames at both training and test time, when they are available, to self-supervise a multi-frame depth estimation system. We show that a straightforward application of self-supervised training to a multi-view plane-sweep stereo architecture produces poor results, significantly worse than self-supervised single frame networks. To overcome this, we introduce several innovations to address issues caused by moving objects, scale ambiguity, and static cameras. We call our resultant multi-frame system **ManyDepth**.

We make the following three contributions:

1. A novel *self-supervised* multi-frame depth estimation model that combines the strengths of monocular and multi-view depth estimation by making use of multiple frames at test time, when they are available.
2. We show that moving objects and static scenes significantly impact self-supervised multi-view matching approaches, and we introduce efficient losses and train-

ing solutions to alleviate this problem.

3. We propose an adaptive cost volume to overcome the scale ambiguity arising from self-supervised training on monocular sequences. To the best of our knowledge, this is the first time cost volume extents have been learned from data rather than set as parameters.

Our ManyDepth model outperforms existing single and multi-frame approaches on the KITTI and Cityscapes datasets.

2. Related work

2.1. Monocular depth estimation

The goal of monocular depth estimation is to predict the depth of each pixel in a single input image. Supervised approaches either make use of dense supervision from depth sensors e.g. [15, 14, 20] or sparse supervision from human annotations e.g. [8]. Self-supervised methods remove the limitation of requiring ground truth depth supervision, instead training with image-reconstruction losses using stereo pairs [87, 21, 23] or monocular video sequences [99].

Recent advances in self-supervised training have focused on addressing various challenges resulting from learning from images alone e.g. more robust image reconstruction losses [26, 72], discrete rather than continuous depth predictions [56, 25, 40], feature space reconstruction losses [93, 72], sparse automatically generated depth supervision [49, 83], occlusion handling [26], improved network architectures [28], and moving objects during training [69, 24, 26, 78, 92, 9, 3, 75, 48, 53] and at test time [54]. Our underlying monocular architecture is based on [24], and could similarly benefit from many of the above enhancements.

2.2. Multi-frame monocular depth estimation

There is a growing number of works that extend existing self-supervised monocular models so that they can leverage temporal information at *test time* to improve the quality of the predicted depth. It is worth noting that there are also several non-deep-learning methods that also aim to produce consistent sequential depth estimates e.g. [96, 44], in addition to conventional SLAM based methods [65, 63, 16, 89], and SLAM methods that integrate a monocular depth estimation network [74, 4, 52]. However, here we focus on state-of-the-art neural network based depth estimation.

Test-time refinement approaches adapt monocular methods to use sequence information at test time e.g. [5, 9, 59, 62, 72, 51]. As self-supervised training does not require any ground truth depth supervision, the same losses used during training can be applied to the test frames to update the network's parameters. The downside is that this necessitates the additional computation of multiple forward and backward model passes for a set of test frames, potentially taking several seconds to perform per set [62, 59].

	Train		Test			
	Needs Depth	Needs Pose	Needs Pose	Object Motion	Single Frame	Multi Frame
Two Frame e.g. [2]	Yes	Yes	No	No	No	Yes
Supervised MVS e.g. [37]	Yes	Yes	Yes	No	No	Yes
Self-sup MVS e.g. [46]	No	Yes	Yes	No	No	Yes
Supervised MD e.g. [20]	Yes	No	No	Yes	Yes	No
Self-sup MD e.g. [99]	No	No	No	Yes	Yes	No
ManyDepth (Ours)	**No**	**No**	**No**	**Yes**	**Yes**	**Yes**

Table 1. Comparison of existing approaches that estimate depth from collections of images. Our approach requires no ground-truth supervision and is robust to object motion. MVS stands for Multi-View Stereo, and MD stands for Monocular Depth.

A second broad group of approaches combine traditional monocular networks with recurrent layers to process sequences of frames e.g. [66, 81, 50, 97]. A related approach uses pairs of sequential frames at test time, sharing features between the pose and depth modules [79] or computing depth-from-flow [88]. These approaches are much more efficient compared to test-time refinement, but they can be more computationally demanding during training due to the need to extract features from multiple frames in a sequence. A further limitation of these methods is that they do not explicitly reason about geometry during inference; they simply rely on the network having learned how to extract meaningful temporal representations.

Our experiments show that our approach often outperforms test-time refinement in terms of accuracy while retaining the efficiency of recurrent methods at inference.

2.3. Deep multi-view depth estimation

Our problem of predicting depth from multiple frames is related to multi-view depth estimation. While early deep stereo methods used mostly convolutional layers to map from images pairs to depth using ground-truth supervision e.g. [61, 77, 55], [45] showed that integrating a plane-sweep stereo cost volume significantly improved results. Recent approaches improved the underlying architectures and contributed more effective ways of regularizing the cost volume [7, 94, 95, 10]. It is also possible to train stereo networks without ground truth supervision [98, 82, 1, 36], but these models are typically outperformed by supervised variants. Some works fuse conventional matching-based stereo estimation with monocular depth cues [71, 60, 17]. In contrast, we do not require stereo pairs during training or testing.

A more general version of the stereo-matching problem is multi-view stereo (MVS), which operates on unordered image collections. Early deep MVS methods used memory-expensive 3D grid representations e.g. [39, 43]. Current supervised approaches, e.g. [37, 80, 91, 38, 57], utilize cost volumes but assume ground truth depth and camera poses

are available for training. They often require camera poses at test time too; which can be refined from an initial estimate [84, 56]. Some methods can predict pose at test time, but they still need to be trained with supervision e.g. [77].

Similar to our approach, [56, 35] process sequences of frames at test time using cost volumes. However, by using ground-truth depth supervision and provided camera poses they side-step the challenges associated with training from self-supervision alone. [86] predict depth from triplets of frames without requiring pose information, but their method cannot deal with variable numbers of frames at test time and is trained with ground truth depth. Concurrent with our work, [85] learn depth from a cost volume, but they use stereo pairs and sparse supervised depth at training time and long sequences for pose estimation.

Most related to us are self-supervised MVS methods that also do not require any ground truth depth i.e. [46, 13]. However, there are several reasons why these existing self-supervised and supervised MVS methods aren't applicable in many scenarios: (i) they need more than one input image at test time, (ii) they assume that the camera is not static, (iii) they typically require camera poses to be provided during training and sometimes also at test time, and (iv) they assume that there are no moving objects in the scene. Our approach leverages the best of monocular and multi-view methods by making use of sequence information at test time, when it is available, while also being robust to scene motion — see Table 1.

3. Problem setup

The aim of depth estimation is to predict a depth map D_t, pixel-aligned with an input image I_t. Conventional single-image depth estimation methods, e.g. [14, 23, 99], train a deep network θ_{depth} to map from I_t to D_t,

$$D_t = \theta_{\text{depth}}(I_t). \tag{1}$$

In contrast, like other *multi-frame* methods [66, 81], our model accepts as input N previous temporal video frames,

$$D_t = \theta_{\text{depth}}(I_t, I_{t-1}, \dots, I_{t-N}). \tag{2}$$

While our model makes use of information from multiple frames, it also can operate in the regime where only one frame is available at test time. Unlike similar works e.g. [50, 97, 9, 59, 5], we do not use *future* frames at test time e.g. I_{t+1}, the use of which would preclude online applications. At training time, we exploit previous *and* future frames as a supervisory signal, and do not require stereo supervision. We do not assume access to the true relative camera pose between I_t and the preceding frames; instead we learn to predict these poses $\{T_n\}_{n=1}^{N}$ at training and test time with a differentiable pose network θ_{pose}, following [99]. We also do not make use of any trained semantic models to mask moving objects e.g. [5, 26, 29]. We do, however, assume known fixed camera intrinsics K — though we could relax these requirements [26, 18].

4. Method

Our method starts with two well established components: self-supervised reprojection based training, and a multi-view cost volume. We then introduce three important innovations that enable cost volume matching to work with self-supervised training from monocular video: (1) adaptive cost volumes, (2) a method to prevent a failure mode we refer to as 'cost volume overfitting', and (3) augmentation for static cameras and single frame inputs.

4.1. Self-supervised monocular depth estimation

Following [99], we train a self-supervision depth network using only video frames that are temporally close to I_t. We use the current estimated depth D_t and the pose network θ_{pose}'s estimate of relative camera pose $T_{t\to t+n}$ to synthesize the scene from the same viewpoint as I_t, but only using pixels from neighboring source frames i.e. $\{I_{t+n}, n \in \{-1, 1\}\}$. The synthesized counterpart to I_t is

$$I_{t+n\to t} = I_{t+n}\Big\langle \text{proj}(D_t, T_{t\to t+n}, K)\Big\rangle, \tag{3}$$

where $\langle\rangle$ is the sampling operator and proj returns the 2D coordinates of the depths in D_t when reprojected into the camera of I_{t+n}. Note that while our cost volume, described later, only uses *preceding* frames to enable online applications, at training-time our reprojection loss uses future frames too. Following [24], for each pixel we optimize the loss for the *best matching source image*, by selecting the per pixel minimum over the reconstruction loss pe,

$$L_p = \min_n pe(I_t, I_{t+n\to t}). \tag{4}$$

We set pe as a combination of SSIM and L_1 losses, and we minimize this loss over all the pixels in the training images over four output scales; see [24] for more details.

4.2. Building a cost volume

To exploit multiple *input* frames, inspired by [11, 42, 45] we build a cost volume which measures the geometric compatibility at different depth values between the pixels from I_t and nearby source frames from the input video. This requires knowledge of relative pose T, which we estimate with the pose network θ_{pose}, trained using a reprojection loss. We define a set of ordered planes $\mathcal{P}$, each perpendicular to the optical axis at I_t and with depths linearly spaced between $d_{\min}$ and $d_{\max}$. Each frame is encoded into a deep feature map F_t and warped to the viewpoint of I_t using each of the hypothesised alternative depths $d \in \mathcal{P}$ using the known camera intrinsics and estimated pose. This creates

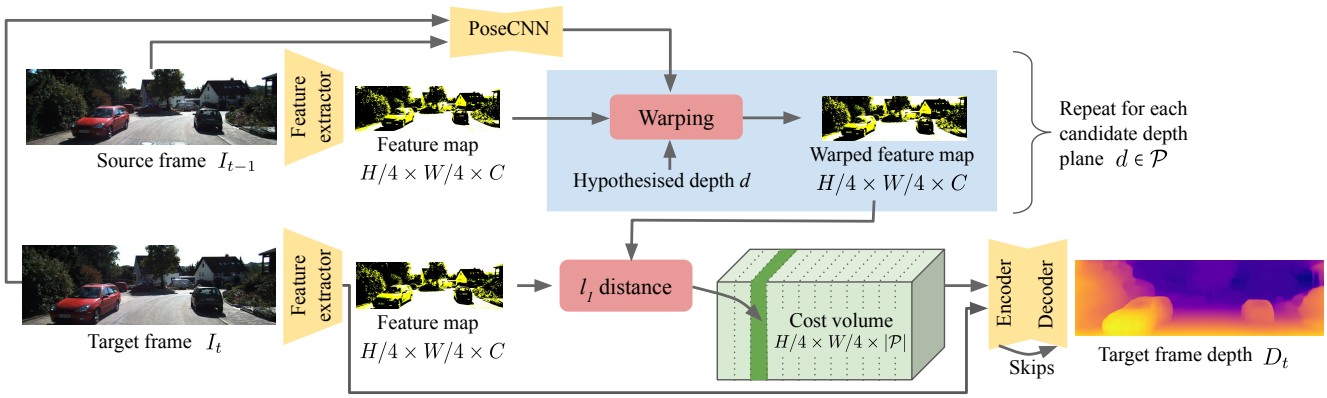

Figure 2. **Our cost volume based depth estimator.** Our depth network θ_{depth} has three main components: a feature extractor, an encoder, and a depth decoder. Our pose network θ_{pose} estimates the relative pose between pairs of images, which is then used to build a cost volume in the reference frame of the target image I_t by warping features extracted from images at different time points. The encoder and depth decoder processes the cost volume to produce a depth image for I_t.

a warped feature map $F_{t+n \to t,d}$. The final cost volume is constructed as the absolute difference between the warped features and the features from I_t, at each $d \in \mathcal{P}$. This is averaged over all source images, following [65]. The cost volume effectively says 'for pixel (i, j), what is the likelihood of the correct depth being d, for each d in $\mathcal{P}$?'. Following [19], the cost volume is concatenated with features F_t and used as input to a convolutional decoder which regresses the depth D_t. See Fig. 2 for an overview.

Cost volumes have the benefit of allowing the network to leverage inputs from multiple viewing angles. However, they typically require $d_{\min}$ and $d_{\max}$ to be chosen as hyperparameters, and they assume that world is static. In the following sections we show how to relax these assumptions.

4.3. Adaptive cost volumes

Cost volume approaches have a problem of needing a known depth range i.e. $d_{\min}$ and $d_{\max}$. These are typically selected as hyperparameters in advance of training based on prior knowledge of the dataset [13] or from known camera poses [37]. We are unable to do this, as self-supervised depth estimation trained on monocular sequences only estimates depth 'up to scale'. This means that while we assume that the final predicted depths, and corresponding poses from the pose network, will all end up in broad agreement with *each other*, they will be different from real-world depth by an unknown scaling factor.

To solve this problem we introduce a novel *adaptive cost volume*, by allowing $d_{\min}$ and $d_{\max}$ to be learned from the data, so they can adjust during training as the network finds its own scaling. This is done using the current predictions from the network of D_t, whereby we compute the average min and max of each D_t over a training batch. These are then used to update an exponential moving average estimates of $d_{\min}$ and $d_{\max}$ with momentum 0.99. $d_{\min}$ and $d_{\max}$ are saved along with the model weights and then kept fixed at test time. Our approach contrasts to [27] who adapt $d_{\min}, d_{\max}$ at *test time* in a coarse-to-fine manner.

4.4. Addressing cost volume overfitting

We observe that our baseline cost volume model trained with monocular supervision suffers from severe artefacts, including large 'holes' punched on moving objects. These are similar to artefacts observed in monocular $I_t \to D_t$ models (see [5, 24] for a description). However, in our cost volume network they are far more severe (see Fig. 3 (c)).

Why does the monocular-trained cost volume fail? In theory, our model should do well. It is trained with a similar reprojection loss used to train state-of-the-art single-frame depth estimators, but it also has access to an additional source of information via the cost volume. However, the information contained in the cost volume is only reliable in specific scenarios e.g. in static regions with textured surfaces. In regions where objects are moving, or where surfaces are untextured, the cost volume will be an unreliable source of depth information (Fig. 3 (b)). For example, the moving car in Fig. 3 results in a match in the cost volume at an incorrect depth and corresponds to a very low reprojection loss. During training, the network becomes over-reliant on the cost volume. Instead of ignoring the cost volume around moving objects, it trusts it too much. Artefacts in the cost volume from moving objects are then introduced in the final depth map, at both training and test time. Ultimately, the final predicted depths *inherit the cost volume's mistakes*. We introduce a method to correct this during training, by teaching the network not to trust the cost volume in these unreliable regions.

Using a separate network to regularize. We make the observation that *single-image* depth networks do not have a cost volume, so are unaffected by 'cost volume overfitting'. While moving objects can still be a problem for these meth-

ods during training [5, 24, 69], in general they make far less severe mistakes on moving objects. We therefore use a monocular network at training time to help 'teach' our cost volume network the right answer — but only in regions we suspect the cost volume to be problematic. This separate network $\theta_{\text{consistency}}$ produces a depth map $\hat{D}_t$ for each training image, and is discarded after training. $\theta_{\text{consistency}}$ shares θ_{pose} with our main network to help ensure scale-consistent predictions between θ_{depth} and $\theta_{\text{consistency}}$. Potentially problematic pixels in our multi-frame output are identified by a binary mask M. In these masked regions we apply an L_1 loss on D_t, encouraging the predictions to be similar to $\hat{D}_t$,

$$L_{\text{consistency}} = \sum M\,|D_t - \hat{D}_t|. \tag{5}$$

Gradients to $\hat{D}_t$ are blocked, ensuring knowledge only transfers from teacher to student and not vice versa.

Identifying unreliable pixels. Our binary mask M is 1 in regions considered to be unreliable, and 0 otherwise. To generate this mask we again make use of $\hat{D}_t$. We reason that in regions where the cost volume is *reliable*, the depth represented by $\hat{D}_t$ will be similar to the depths represented by the argmin of the cost volume. We therefore compare the depth represented by the argmin of the cost volume (i.e. D_{cv}, not D_t) to the depth $\hat{D}_t$ predicted by $\theta_{\text{consistency}}$. The mask M is set to 1 only in regions where $\hat{D}_t$ and D_{cv} differ significantly, so

$$M = \max\left(\frac{D_{\text{cv}} - \hat{D}_t}{\hat{D}_t}, \frac{\hat{D}_t - D_{\text{cv}}}{D_{\text{cv}}}\right) > 1. \tag{6}$$

The idea of using a separate 'disposable' network to help to regularize training is not new, e.g. [69, 99]. Our novelty is in using a *single-frame* depth network to improve a *multi-frame* system. Our approach is also less costly and less constrained than using offline semantic segmentation [26], and makes fewer assumptions than RANSAC-based filtering [29]. In our experiments we compare to two alternative masking schemes from [69] and [24], and show that our approach is superior.

4.5. Static cameras and start-of-sequences

Using multiple frames at test time introduces two potential challenges for our method. The first issue is when I_{t-1} does not exist, i.e. when predicting depth for a single image or those at the start of a sequence. This case is trivially handled by monocular methods as they only require single frames as input. However, MVS approaches fail in these situations. To address this problem, during training with probability p, we replace the cost volume with a tensor of zeros. For these images, this encourages the network to learn to rely only on features directly from I_t. Then at test time, when I_{t-1} does not exist, we simply replace the cost volume with zeros. The second case arises when the camera does not move between I_{t-1} and I_t, e.g. a car stopped at traffic lights. Again this is another failure case for MVS methods. To address this at training time, with probability q, we replace the I_{t-1} input to the cost volume with a color-augmented version of I_t, but still supervise with the 'real' I_{t-1}, I_{t+1} in Eqn. 4. This enables the network to predict plausible depths even when the cost volume is constructed from images with no camera baseline.

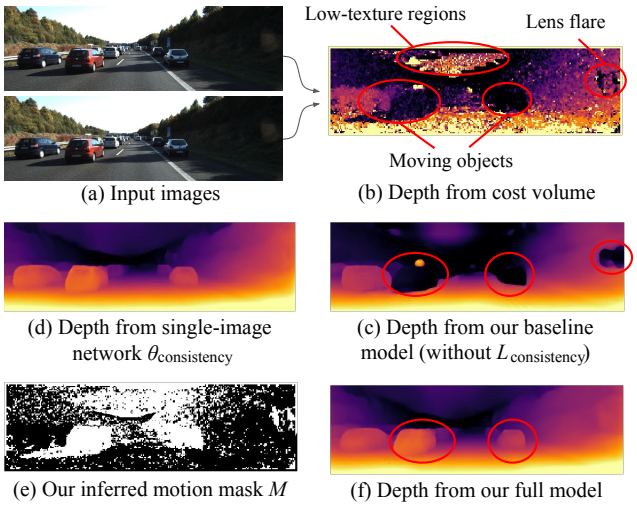

Figure 3. **We enable self-supervised training to work with a cost volume.** When we build a cost volume from a sequence (a), moving objects create incorrect depths in the cost volume (b). These errors are propagated through to our predicted depth maps (c). 'Traditional' single-image depth estimation does not exhibit this failure mode, but can still produce biased depth (d). We therefore use a single-image network to help our model recover from this failure mode, but only on pixels identified to be unreliable by our mask M (e). Our final prediction (f) is superior to both our baseline and the single-image network.

Our final loss is $L = (1 - M)L_p + L_{\text{consistency}} + L_{\text{smooth}}$, where L_{smooth} is the smoothness loss from [23].

5. Implementation details

We use training-time color and flip augmentations on images being fed to the depth and pose networks, using the settings from [24]. Unless otherwise stated, all our models are trained with an input and output resolution of 640×192, and we fix $N = 1$, so the cost volume is constructed with frames $\{I_t, I_{t-1}\}$, at both training and test time. In all cases self-supervision during training is from frames $\{I_{t-1}, I_t, I_{t+1}\}$. We train with Adam [47] for 20 epochs with a learning rate of 10^{-4}, dropping by a factor of 10 for the final 5 epochs. After Q epochs, we fix d_{min} and d_{max} and the weights of θ_{pose} and $\theta_{\text{consistency}}$. This allows θ_{depth} to finetune with a non-moving target. We set $Q = 15$ for KITTI, and $Q = 5$ for Cityscapes to account for the larger number of images

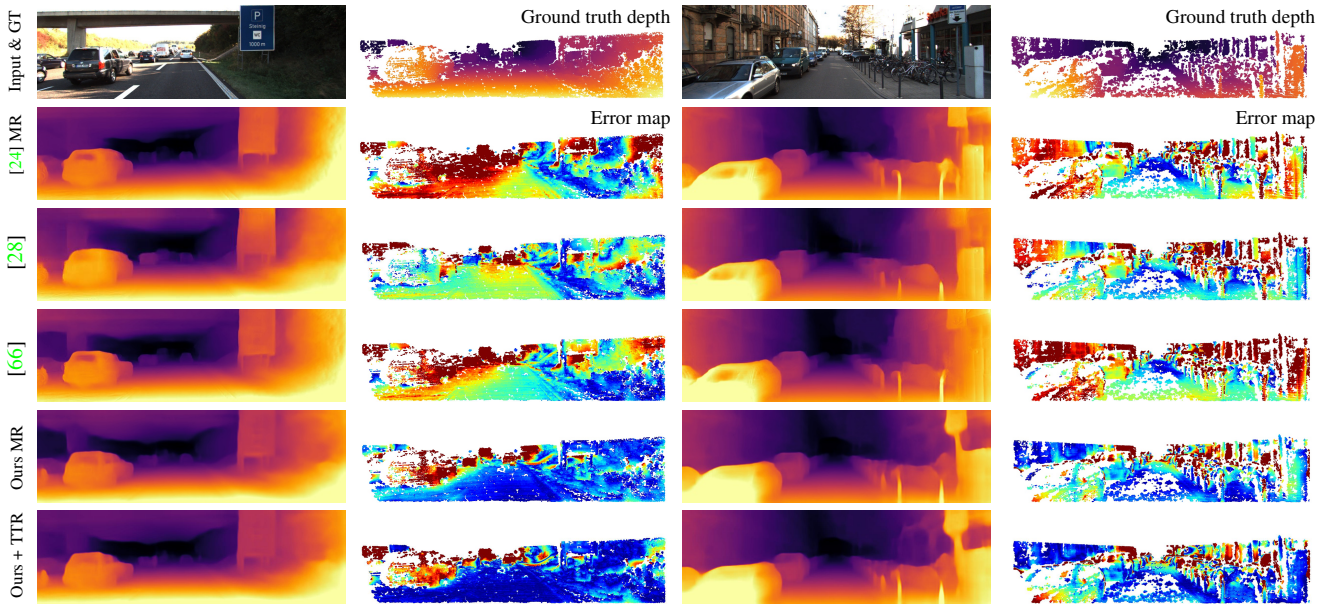

Figure 4. **Qualitative results on KITTI.** Error maps in columns 2 and 4 show the abs. rel. error compared to new ground truth [76], from good (blue) to bad (red). All error maps are colormapped equivalently. While depth maps look qualitatively similar between our multi-frame predictions (bottom rows) and the baselines, the error maps reveal the large 'hidden mistakes' made by methods which only have access to a single test-time image. This is particularly apparent in ambiguous regions, e.g. in the dark embankment on the right of the freeway. Additionally, note that [66] also has access to multiple frames at test time, however their method does not explicitly utilize geometry. This results in an improvement over the single frame [24], but there is noticeably higher error than for our approach.

in the Cityscapes training set. The feature extractor in θ_{depth} comprises the first five ResNet18 layers [33]. These features are aggregated into a cost volume, the result of which is concatenated with our input image features, and followed by the remaining ResNet18 convolutional layers. We use the depth decoder from [24]. Our pose network θ_{pose} and skip connections for θ_{depth} are the same as [24]. $\theta_{\text{consistency}}$ uses the standard architecture from [24] with no modifications. Full architecture details are in the supplementary. Following [24, 83, 79, 31], we use weights pretrained on ImageNet [70], but provide results trained from scratch in the supplementary material. For all our experiments we set $p = q = 0.25$ during training.

6. Experiments

Here we evaluate our ManyDepth model and (1) show that it gives SOTA results by comparing, in a standardized way, to both single-frame and multi-frame depth estimation and (2) validate our design decisions via ablations. Additional results are provided in the supplementary material.

We evaluate on two challenging depth estimation datasets, both of which exhibit moving objects. For both, we use the standard depth evaluation metrics from [14, 15].
(a) KITTI [22]. We use the Eigen split from [14]. This is commonly used for single frame depth estimation, but is more recently also used for multi-frame approaches e.g. [81, 66]. 22 frames in the KITTI Eigen test set are at the start of a sequence, and do not have a previous frame. We still include these images in the evaluation. For these images, the network does not have access to any other frames and thus makes a prediction based on one frame only. In the supplementary material, we additionally include models evaluated on the improved KITTI ground truth [76].
(b) Cityscapes [12]. Following [99, 90, 92], we train on 69,731 images from the monocular sequences, which we preprocess into triples using the scripts from [99]. We do not use stereo pairs or semantics. We evaluate on the 1,525 test images using the provided SGM [34] disparity maps. As with KITTI, we clip predicted depths at 80m, and only evaluate on ground truth depths less than 80m.

6.1. KITTI results

In Table 2 we compare to multi-frame approaches, some of which, e.g. [9, 5, 66, 79, 62], see more frames than ours or also use future frames e.g. [9, 5, 62]. We do not include results from [56] as they do not provide their scores on the KITTI Eigen split (see [88]). We additionally compare to the best-performing self-supervised monocular depth estimation approaches. To control for resolution, we separate low and high resolution models, and we also split methods which use expensive multi-pass test-time refinement into separate sections. We observe that our approach outperforms all previously published self-supervised methods that do not use semantic supervision on most metrics. We also

	TTR	Method	Test frames	Semantics	WxH	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Low and medium resolution		Ranjan *et al.* [69]	1		832 x 256	0.148	1.149	5.464	0.226	0.815	0.935	0.973
		EPC++ [58]	1		832 x 256	0.141	1.029	5.350	0.216	0.816	0.941	0.976
		Struct2depth (M) [5]	1	•	416 x 128	0.141	1.026	5.291	0.215	0.816	0.945	0.979
		Videos in the wild [26]	1	•	416 x 128	0.128	0.959	5.230	0.212	0.845	0.947	0.976
		Guizilini *et al.* [29]	1	•	640 x 192	0.102	**0.698**	**4.381**	0.178	0.896	0.964	**0.984**
		Johnston *et al.* [40]	1		640 x 192	0.106	0.861	4.699	0.185	0.889	0.962	0.982
		Monodepth2 [24]	1		640 x 192	0.115	0.903	4.863	0.193	0.877	0.959	0.981
		Packnet-SFM [28]	1		640 x 192	0.111	0.785	4.601	0.189	0.878	0.960	0.982
		Li *et al.* [53]	1		416 x 128	0.130	0.950	5.138	0.209	0.843	0.948	0.978
		Patil *et al.* [66]	N		640 x 192	0.111	0.821	4.650	0.187	0.883	0.961	0.982
		Wang *et al.* [79]	2 (-1, 0)		640 x 192	0.106	0.799	4.662	0.187	0.889	0.961	0.982
		ManyDepth (MR)	2 (-1, 0)		640 x 192	**0.098**	0.770	4.459	**0.176**	**0.900**	**0.965**	0.983
	•	GLNet [9]	3 (-1, 0, +1)		416 x 128	0.099	0.796	4.743	0.186	0.884	0.955	0.979
	•	Luo *et al.* [59]	N		384 x 112	0.130	2.086	4.876	0.205	0.878	0.946	0.970
	•	CoMoDA [51]	N	•	640 x 192	0.103	0.862	4.594	0.183	0.899	0.961	0.981
	•	McCraith *et al.* [62]	2 (0, +1)		640 x 192	**0.089**	0.747	4.275	0.173	0.912	0.964	0.982
	•	Struct2depth (M+R) [5]	3 (-1, 0, +1)	•	416 x 128	0.109	0.825	4.750	0.187	0.874	0.958	**0.983**
	•	**ManyDepth (MR + TTR)**	2 (-1, 0)		640 x 192	0.090	**0.713**	**4.261**	**0.170**	**0.914**	**0.966**	**0.983**
High resolution		Monodepth2 [24]	1		1024 x 320	0.115	0.882	4.701	0.190	0.879	0.961	0.982
		Packnet-SFM [28]	1		1280 x 384	0.107	0.802	4.538	0.186	0.889	0.962	0.981
		Guizilini *et al.* [29]	1	•	1280 x 384	0.100	0.761	4.270	0.175	0.902	0.965	0.982
		Shu *et al.* [72] (ResNet50)	1		1024 x 320	0.104	0.729	4.481	0.179	0.893	0.965	**0.984**
		Wang *et al.* [79]	2 (-1, 0)		1024 x 320	0.106	0.773	4.491	0.185	0.890	0.962	0.982
		ManyDepth (HR)	2 (-1, 0)		1024 x 320	0.093	0.715	**4.245**	0.172	0.909	0.966	0.983
		ManyDepth (HR ResNet50)	2 (-1, 0)		1024 x 320	**0.091**	**0.694**	**4.245**	**0.171**	**0.911**	**0.968**	0.983
	•	McCraith *et al.* [62]	2 (0, +1)		1024 x 320	0.089	0.756	4.228	0.170	0.917	0.967	**0.983**
	•	Shu *et al.* [72] (ResNet50)	3 (-1, 0, +1)		1024 x 320	0.088	0.712	**4.137**	0.169	0.915	0.965	0.982
	•	**ManyDepth (HR + TTR)**	2 (-1, 0)		1024 x 320	**0.087**	0.696	4.183	**0.167**	0.918	**0.968**	**0.983**
	•	**ManyDepth (HR R50 + TTR)**	2 (-1, 0)		1024 x 320	**0.087**	**0.685**	4.142	**0.167**	**0.920**	**0.968**	**0.983**

Table 2. **Comparison of our method to existing self-supervised approaches on the KITTI [22] Eigen split.** At the top we compare medium and low resolution results without and with test-time refinement (TTR). At bottom we compare high resolution results without and with TTR. The best results in each subsection are in **bold**; second best are underlined. Our method outperforms all previous methods in all subsections across most metrics, whether or not the baselines use multiple frames at test time. We indicate if a method uses semantic supervision (Semantics) and methods indicated by N take a long sequence of frames as input for each test image (e.g. the preceding frames or frames before and after in time).

implement the test-time refinement scheme of [62] on our model, updating the weights of the depth and pose encoders using sequential pairs of images from the test set, for 50 steps. Not surprisingly, this further improves our results, and we outperform other test-time refinement methods.

Qualitative results are presented in Fig. 4. In some cases the predicted depth maps looks qualitatively similar to the monocular only models, but the error maps show the high magnitude of mistakes which can be present.

Efficiency comparison. Fig. 5 illustrates the runtime efficiency of our ManyDepth models (640 x 192: ○, ◇ and 1024 x 320: ●, ◆) compared to other methods, including test-time refinement approaches (×). We report multiply-add computations (MACs) for each method and show that test-time refinement models which perform multiple forward-backward passes are too computationally demanding for use in real-time applications. The supplementary material contains a full results table and additional details.

6.2. KITTI ablation

In Table 4 we show the importance of the various components of our approach by turning them on and off in turn.

ManyDepth w/o motion masking: We omit $L_{\text{consistency}}$ from our loss and set M to zeros everywhere.

ManyDepth w/o motion masking, w/o augmentation: As above, but also omitting our augmentations.

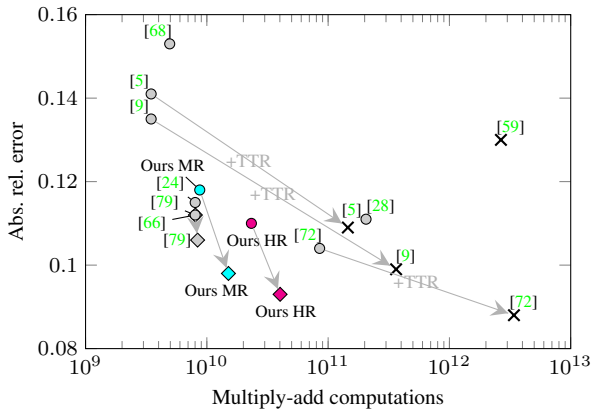

Figure 5. **Our single-pass network is significantly more efficient than test-time optimisation.** We compare abs. rel. error (y-axis) against MACs (x-axis) on the KITTI Eigen test set.
○ – Single frame models. These tend to have low MACs.
× – Multi-frame models which use test-time refinement.
◇ – Multi-frame models with a single forward pass at test time.
Methods which are more accurate than ours take over two orders of magnitude more time to compute (note the logarithmic scale on the x-axis.). Our MR multi-frame version (◇) has better accuracy than [79], who has a similar runtime.

Method	Test frames	Semantics	WxH	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Struct2Depth 2 [6]	1	•	416 x 128	0.145	1.737	7.280	0.205	0.813	0.942	0.976
Pilzer *et al.* [67]	1		512 x 256	0.240	4.264	8.049	0.334	0.710	0.871	0.937
Monodepth2† [24]	1		416 x 128	0.129	1.569	6.876	0.187	0.849	0.957	0.983
Videos in the Wild [26]	1	•	416 x 128	0.127	1.330	6.960	0.195	0.830	0.947	0.981
Li *et al.* [53]	1		416 x 128	0.119	1.290	6.980	0.190	0.846	0.952	0.982
Struct2Depth 2 [6]	3 (-1, 0, +1)		416 x 128	0.222	5.737	8.613	0.258	0.774	0.908	0.954
Struct2Depth 2 [6]	3 (-1, 0, +1)	•	416 x 128	0.151	2.492	7.024	0.202	0.826	0.937	0.972
ManyDepth	2 (-1, 0)		416 x 128	**0.114**	**1.193**	**6.223**	**0.170**	**0.875**	**0.967**	**0.989**

Table 3. **Results on Cityscapes.** Our method gives superior performance to all competing models. † is trained by us with the authors' code, with preprocessing from [99]. Results from [67] are their 'Half-Cycle Mono' model, their only variant not requiring test-time stereo pairs. We evaluate using the cropping scheme of [6] following conversations with the authors; see supplementary material for specifics.

ManyDepth with motion masking but no teacher: We remove $L_{\text{consistency}}$ but still use M to mask L_p.

Stack of 2 frames as input: A baseline which directly maps (I_{t-1}, I_t) to D_t. We modify [24]'s network to accept two images as input, and train using their loss.

ManyDepth with motion masking from [69]: We use our full loss, but our mask M is the same as [69]. We use their pretrained models to compute these masks offline for the entire training set.

ManyDepth with motion masking from [24]: Here we use our full loss, but set the mask M to an 'automask' from [24].

Khot *et al.* [46]: We trained this unsupervised MVS approach on KITTI, with the implementation from [30].

ManyDepth (I_{t-2}, I_{t-1}, I_t) and ManyDepth (I_{t-1}, I_t, I_{t+1}): Retrained variants of our model which build the cost volume from three frames instead of just two. This improves some metrics but not all.

Benefit of our augmentations. In Table 5 we evaluate three different scenarios, comparing our model to a baseline which was trained without our augmentations from Section 4.5. When evaluating in 'standard' mode (i.e. using the previous and current frames as input) on the entire KITTI test set, the difference between the two models is negligible. This is partially because the KITTI test images are predominately from a moving camera. However, when we evaluate in 'start-of-sequence' mode (i.e. the standard monocular setting using only (I_t) as input) and 'static camera' evaluation mode (i.e. simulating a static camera with inputs (I_t, I_t)), our augmentation scheme is significantly better.

6.3. Cityscapes results

In Table 3 we perform additional comparisons where we train and test on the Cityscapes dataset [12]. Again, we consistently outperform competing methods, even those that use semantic supervision.

7. Conclusion

We presented a fully self-supervised online method that predicts superior depths from a single image, or from multiple images when they are available. We achieve the benefits of both multi-frame and monocular methods, while being more robust on moving objects and static cameras compared to a naive integration of a cost volume. We presented state-of-the-art results on both the KITTI and Cityscapes datasets. While test-time refinement methods are close competitors in terms of depth accuracy, we have shown that our method is significantly more efficient during inference. We expect that our results could be further improved via recent complimentary advances in monocular depth estimation e.g. discretized output depths [25] or feature based losses [72].

Ablation	Abs Rel	Sq Rel	RMSE
ManyDepth full	**0.098**	**0.770**	4.459
ManyDepth (w/o motion masking)	0.113	1.354	5.228
ManyDepth (w/o motion masking, w/o aug.)	0.284	11.240	8.516
ManyDepth (with motion masking, w/o teacher)	0.154	2.682	6.573
Stack of 2 frames as input (I_{t-1}, I_t)	0.121	1.028	5.016
ManyDepth (with motion masking from [69])	0.099	0.783	**4.447**
ManyDepth (with motion masking from [24])	0.099	0.780	4.465
Khot *et al.* [46] reimplementation	0.200	4.694	7.232
ManyDepth with 3-frame input (I_{t-2}, I_{t-1}, I_t)	0.098	0.780	**4.430**
ManyDepth with 3-frame input (I_{t-1}, I_t, I_{t+1})	**0.097**	**0.768**	4.431

Table 4. **Our contributions lead to better scores.** Here we ablate our ManyDepth method on KITTI 2015 [22] using the Eigen split. Full numbers are in the supplementary material.

Test-time input	Model	Abs Rel	Sq Rel	RMSE
Standard: (I_{t-1}, I_t)	No augmentation	0.100	0.794	**4.432**
	ManyDepth	**0.098**	**0.770**	4.459
Start-of-sequence: (I_t)	Monodepth2 [24]	**0.115**	0.903	4.863
	No augmentation	0.148	1.076	5.161
	ManyDepth	0.118	**0.892**	**4.764**
Static camera: (I_t, I_t)	No augmentation	0.158	1.132	5.228
	ManyDepth	**0.117**	**0.886**	**4.754**

Table 5. **Our augmentations help in static camera and start-of-sequence cases.** We compare two variants of our model, one trained with our novel data augmentations ('Ours') and one without. We create two artificial scenarios to test each model's performance on start-of-sequence images (where we just input I_t) and static cameras (where both input frames are the exact same).

Acknowledgements. Thanks to Jamie Shotton; a conversation with him at a BMVA workshop motivated work in this area. Also to Daniyar Turmukhambetov and Sara Vicente for their valuable feedback, and to the authors of [6] for CityScapes evaluation help.

References

[1] Filippo Aleotti, Fabio Tosi, Li Zhang, Matteo Poggi, and Stefano Mattoccia. Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation. In *ECCV*, 2020.

[2] V Madhu Babu, Kaushik Das, Anima Majumdar, and Swagat Kumar. Undemon: Unsupervised deep network for depth and ego-motion estimation. In *IROS*, 2018.

[3] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, 2019.

[4] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM—learning a compact, optimisable representation for dense visual SLAM. In *CVPR*, 2018.

[5] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019.

[6] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised monocular depth and ego-motion learning with structure and semantics. In *CVPR Workshops*, 2019.

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[8] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016.

[9] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[10] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *PAMI*, 2019.

[11] Robert T Collins. A space-sweep approach to true multi-image matching. In *CVPR*, 1996.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[13] Yuchao Dai, Zhidong Zhu, Zhibo Rao, and Bo Li. MVS^2: Deep unsupervised multi-view stereo with multi-view symmetry. In *3DV*, 2019.

[14] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[15] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

[16] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular slam. In *ECCV*, 2014.

[17] José M Fácil, Alejo Concha, Luis Montesano, and Javier Civera. Single-view and multi-view depth fusion. *IEEE Robotics and Automation Letters*, 2017.

[18] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. CAM-Convs: Camera-aware multi-scale convolutions for single-view depth. In *CVPR*, 2019.

[19] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[20] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[21] Ravi Garg, Vijay Kumar BG, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[23] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[24] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[25] Juan Luis Gonzalez and Munchurl Kim. Forget about the LiDAR: Self-supervised depth estimators with MED probability volumes. *NeurIPS*, 2020.

[26] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *ICCV*, 2019.

[27] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020.

[28] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *CVPR*, 2020.

[29] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *ICLR*, 2020.

[30] Xiaoyang Guo. PyTorch implementation of MVS-Net. https://github.com/xy-guo/MVSNet_pytorch, 2020.

[31] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. Learning monocular depth by distilling cross-domain stereo networks. In *ECCV*, 2018.

[32] Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. High-quality depth from uncalibrated small motion clip. In *CVPR*, 2016.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[34] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *PAMI*, 2007.

[35] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *ICCV*, 2019.

[36] Baichuan Huang, Can Huang, Yijia He, Jingbin Liu, and Xiao Liu. M^{3}VSNet: Unsupervised multi-metric multi-view stereo network. *arXiv:2005.00363*, 2020.

[37] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018.

[38] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end deep plane sweep stereo. *ICLR*, 2019.

[39] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *ICCV*, 2017.

[40] Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, 2020.

[41] Neel Joshi and C Lawrence Zitnick. Micro-baseline stereo. *Microsoft Research Technical Report*, 2014.

[42] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. Handling occlusions in dense multi-view stereo. In *CVPR*, 2001.

[43] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017.

[44] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *PAMI*, 2014.

[45] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017.

[46] Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning unsupervised multi-view stereopsis via robust photometric consistency. In *CVPR Workshops*, 2019.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[48] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *ECCV*, 2020.

[49] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning SFM from SFM. In *ECCV*, 2018.

[50] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. DepthNet: A recurrent neural network architecture for monocular depth prediction. In *CVPR Workshops*, 2018.

[51] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. CoMoDA: Continuous monocular depth adaptation using past experiences. In *WACV*, 2021.

[52] Tristan Laidlow, Jan Czarnowski, and Stefan Leutenegger. DeepFusion: real-time dense 3D reconstruction for monocular SLAM using single-view depth and gradient predictions. In *ICRA*, 2019.

[53] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Unsupervised monocular depth learning in dynamic scenes. In *CoRL*, 2020.

[54] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T. Freeman. Learning the depths of moving people by watching frozen people. In *CVPR*, 2019.

[55] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *CVPR*, 2018.

[56] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural RGB→D sensing: Depth and uncertainty from a video camera. In *CVPR*, 2019.

[57] Xiaoxiao Long, Lingjie Liu, Christian Theobalt, and Wenping Wang. Occlusion-aware depth estimation with adaptive normal constraints. In *ECCV*, 2020.

[58] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding. *PAMI*, 2019.

[59] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. In *ACM SIGGRAPH*, 2020.

[60] Diogo Martins, Kevin Van Hecke, and Guido De Croon. Fusion of stereo and still monocular depth estimates in a self-supervised learning context. In *ICRA*, 2018.

[61] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[62] Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Monocular depth estimation with self-supervised instance adaptation. *arXiv:2004.05821*, 2020.

[63] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In *CVPR*, 2010.

[64] Richard A Newcombe, Shahram Izadi, and Otmar Hilliges. Kinectfusion: Real-time dense surface mapping and tracking. In *UIST*, 2011.

[65] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011.

[66] Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. Don't forget the past: Recurrent depth estimation from monocular video. In *IEEE Robotics and Automation Letters*, 2020.

[67] Andrea Pilzer, Dan Xu, Mihai Marian Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *3DV*, 2018.

[68] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on CPU. In *IROS*, 2018.

[69] Anurag Ranjan, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, 2019.

[70] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[71] Ashutosh Saxena, Jamie Schulte, and Andrew Y Ng. Depth estimation using monocular and stereo cues. In *IJCAI*, 2007.

[72] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *ECCV*, 2020.

[73] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *CVPR Workshops*, 2018.

[74] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, 2017.

[75] Fabio Tosi, Filippo Aleotti, Pierluigi Zama Ramirez, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *CVPR*, 2020.

[76] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *3DV*, 2017.

[77] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *CVPR*, 2017.

[78] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. SfM-Net: Learning of structure and motion from video. *arXiv:1704.07804*, 2017.

[79] Jianrong Wang, Ge Zhang, Zhenyu Wu, XueWei Li, and Li Liu. Self-supervised joint learning framework of depth estimation via implicit cues. *arXiv:2006.09876*, 2020.

[80] Kaixuan Wang and Shaojie Shen. MVDepthNet: Real-time multiview depth estimation neural network. In *3DV*, 2018.

[81] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-)supervised learning of monocular video visual odometry and depth. In *CVPR*, 2019.

[82] Yang Wang, Peng Wang, Zhenheng Yang, Chenxu Luo, Yi Yang, and Wei Xu. UnOS: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *CVPR*, 2019.

[83] Jamie Watson, Michael Firman, Gabriel Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[84] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. DeepSFM: Structure from motion via deep bundle adjustment. In *ECCV*, 2020.

[85] Felix Wimbauer, Nan Yang, Lukas von Stumberg, Niclas Zeller, and Daniel Cremers. MonoRec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. In *CVPR*, 2021.

[86] Zhenyao Wu, Xinyi Wu, Xiaoping Zhang, Song Wang, and Lili Ju. Spatial correspondence with generative adversarial network: Learning depth from monocular videos. In *ICCV*, 2019.

[87] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *ECCV*, 2016.

[88] Jiaxin Xie, Chenyang Lei, Zhuwen Li, Li Erran Li, and Qifeng Chen. Video depth estimation by fusing flow-to-depth proposals. In *IROS*, 2020.

[89] Luwei Yang, Feitong Tan, Ao Li, Zhaopeng Cui, Yasutaka Furukawa, and Ping Tan. Polarimetric dense monocular SLAM. In *CVPR*, 2018.

[90] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. LEGO: Learning edge with geometry all at once by watching videos. In *CVPR*, 2018.

[91] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[92] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018.

[93] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.

[94] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, 2019.

[95] Feihu Zhang, Xiaojuan Qi, Ruigang Yang, Victor Prisacariu, Benjamin Wah, and Philip Torr. Domain-invariant stereo matching networks. In *ECCV*, 2020.

[96] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Consistent depth maps recovery from a video sequence. *PAMI*, 2009.

[97] Haokui Zhang, Chunhua Shen, Ying Li, Yuanzhouhan Cao, Yu Liu, and Youliang Yan. Exploiting temporal consistency for real-time video depth estimation. In *ICCV*, 2019.

[98] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *arXiv:1709.00930*, 2017.

[99] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.